AF

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Steven R. CHALMER, et al.	:	
Appl. No.: 09/605,812	:	Art Unit: 2195
Filed: June 28, 2000	:	Examiner: TO, Jennifer N.
For: **REPLACEABLE SCHEDULING ALGORITHM IN MULTITASKING KERNEL**	:	Atty Docket: EMS-00801

CERTIFICATE OF MAILING

I hereby certify that the foregoing document is being deposited with the United States Postal Service as first class mail, postage prepaid, "Post Office to Addressee", in an envelope addressed to: Mail Stop APPEAL BRIEF-PATENTS, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on May 31, 2007.

Bonny Rogers

RESPONSE TO NOTICE OF NON-COMPLIANT APPEAL BRIEF

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In response to the Notice of Non-Compliant Appeal Brief dated May 21, 2007, that was received for the above-referenced application, Applicants submit the attached amended Appeal Brief prepared in accordance with the guidelines set forth in the Notice and in compliance with 37 C.F.R. 41.37 and to update the case law to reflect with recent decisions by the U.S. Supreme Court.

Should there be any questions after reviewing this paper, the Examiner is invited to contact the undersigned at 508-898-8603. No fees are believed to be due in this matter. However, if we are mistaken, please charge any fees that may be associated with this response to our Deposit Account No. 503596.

Respectfully submitted,
MUIRHEAD AND SATURNELLI, LLC

Date: May 31, 2007

Donald W. Muirhead
Registration No. 33,978

Muirhead and Saturnelli, LLC
200 Friberg Parkway, Suite 1001
Westborough, MA 01581
(508) 898-8601

Customer No.: 52427

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

CERTIFICATE OF MAILING

I hereby certify that the foregoing document is being deposited with the United States Postal Service as first class mail, postage prepaid, "Post Office to Addressee", in an envelope addressed to: Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA, 22313-1450 on May 31, 2007.

Bonny J. Rogers

* * *

APPEAL BRIEF UNDER 37 C.F.R. § 41.37

(AMENDED)

Application Serial No.: 09/605,812

Filed: June 28, 2000

Applicants/Appellants: Steven R. CHALMER, et al.

Title: REPLACEABLE SCHEDULING ALGORITHM IN MULTITASKING KERNEL

Appeal from a decision of the Primary Examiner
dated October 13, 2006
and Advisory Action dated January 17, 2007

Atty. Docket: EMS-00801

REAL PARTY IN INTEREST

The above-identified application is assigned to EMC Corporation by virtue of an Assignment recorded by the U.S. Patent and Trademark Office on June 28, 2000, at Reel 010938, Frame 0345.

RELATED APPEALS AND INTERFERENCES

Appellants are not aware of any appeals or interferences related to the above identified application.

STATUS OF CLAIMS

This is an appeal from a decision of the Primary Examiner in the Final Office Action dated October 13, 2006, finally rejecting Claims 1-3, 5-14, 16-20, 22-31, 33 and 34 in the above-identified patent application and from the Advisory Action dated January 17, 2007 containing additional comments on the rejections. Claims 1-3, 5-14, 16-20, 22-31, 33 and 34 stand rejected under 35 U.S.C. 103(a). Claims 4, 15, 21, and 32 have been previously cancelled. No claim has been allowed. A Notice of Appeal was submitted on February 9, 2007, for all pending claims.

STATUS OF AMENDMENTS

Appellants filed an After-Final Response on December 26, 2006, in which no claim amendments were made. Therefore, all amendments proposed by the Appellants throughout previous prosecution of this case have been entered. The claims involved in this Appeal are set forth in the attached Claims Appendix.

SUMMARY OF CLAIMED SUBJECT MATTER

Prior to discussing the claimed invention specifically, it may be useful to briefly provide general information to aid in the understanding of the claimed invention.

A scheduler is a special type of operating system process that manages the running of other processes on a processor. The other processes may be, for example, user applications or other operating system processes. In the simplest case, a single scheduler schedules a plurality of processes to run on a single processor. The scheduler may use a particular algorithm to determine which processes run, in what order, and for how long. For example, the scheduler may use a round robin technique where each process that is available to run is scheduled one at a time and, after each process has been scheduled once, each process is then rescheduled, etc. There are other scheduling algorithms that use different algorithms to determine which process runs, in what order, and for how long.

The presently-claimed invention provides a system that has a plurality of schedulers for a multitasking system for a processor and in which control may be switched from a first scheduler to a second scheduler during run time. Note that this may be contrasted from a plurality of processes that are scheduled by the scheduler and the claims specifically recite that the scheduler is different from the processes being scheduled by the scheduler. The particular scheduler that is chosen for use in switching from a first scheduler to a second scheduler during run time depends upon run time conditions. (See, for example, pages 22, line 9 to page 23, line 22 of the originally-filed specification.)

Turning specifically to the claims, independent claim 1 recites a method of switching during run time from a first scheduler to a second scheduler for a multitasking system for a processor (e.g., see FIG. 10, FIG. 2 and page 22, line 18 to page 23, line 14). The method is recited as including choosing the second scheduler (for example, the statistical code profiler, see page 23, lines 10-12) from a plurality of schedulers (e.g., other forms of schedulers, see page 23, line 15), where at least one of the plurality of schedulers (e.g., profiler as scheduler 32, see FIG. 2) selects processes to be run from a plurality of runable processes (e.g., 34-36, see FIG. 2) different from the plurality of schedulers and where choosing the second scheduler (e.g., profiler) is based on parameters that vary according to run time conditions (e.g., see page 22, lines 20-22 and page 23, lines 18-21). The method is also recited as including setting, during a context switch operation, a program counter to an address corresponding to code of the second scheduler and the processor executing code of the second scheduler at an address corresponding to the program counter (e.g., see page 23, lines 10-14; see also FIG. 4 and page 12, lines 7-14). Claims 2, 3, and 5-8 depend directly or indirectly from independent claim 1.

Independent claim 9 recites a method of scheduling tasks in a multitasking operating system (e.g., see FIG. 10, FIG. 2 and page 22, line 18 to page 23, line 14). The method includes using a first scheduler to schedule tasks (e.g., current/original scheduler, see page 23, lines 9-14), choosing a second scheduler (e.g., profiler, see page 23, lines 9-14) from a plurality of schedulers (e.g., other forms of schedulers, see page 23, line 15), where at least one of the plurality of schedulers (e.g., profiler as scheduler 32, see FIG. 2) selects processes (e.g., 34-36, see FIG. 2) to be run from a plurality of runable processes different from the plurality of schedulers and where choosing the second scheduler (e.g., profiler) is based on

parameters that vary according to run time conditions (e.g., see page 22, lines 20-22 and page 23, lines 18-21). The method also recites switching, during run time, from using the first scheduler to schedule tasks to using the second scheduler to schedule tasks (e.g., see page 22, lines 18-22). Claims 10-14, 16, and 17 depend directly or indirectly from independent claim 9.

Independent claim 18 recites computer software in combination with a computer readable medium (e.g, see page 4, lines 3-8), that switches, during run time, from a first scheduler to a second scheduler for a multitasking system for a processor (e.g., see page 22, lines 18-22). The software is recited as including executable code, provided on a computer readable medium, that chooses the second scheduler (e.g., profiler, see page 23, lines 9-14) from a plurality of schedulers (e.g., other forms of schedulers, see page 23, line 15), where at least one of the plurality of schedulers (e.g., profiler as scheduler 32, see FIG. 2) selects processes (e.g., 34-36, see FIG. 2) to be run from a plurality of runable processes different from the plurality of schedulers and where executable code that chooses the second scheduler uses parameters that vary according to run time conditions (e.g., see page 22, lines 20-22 and page 23, lines 18-21). The software is also recited as including executable code, provided on a computer readable medium, that sets a program counter to an address corresponding to code of the second scheduler and executable code, provided on a computer readable medium, that causes the processor to execute code at an address corresponding to the program counter (e.g., see page 23, lines 10-14; see also FIG. 4 and page 12, lines 7-14). Claims 19, 20 and 22-25 depend directly or indirectly from independent claim 18.

Independent claim 26 recites computer software in combination with a computer readable medium (e.g, see page 4, lines 3-8) that schedules tasks in a multitasking operating system (e.g., see page 22, lines 18-22). The software is recited as including executable code, provided on a computer readable medium, that uses a first scheduler to schedule tasks (e.g., current/original scheduler, see page 23, lines 9-14), executable code, provided on a computer readable medium, that chooses a second scheduler (e.g., profiler, see page 23, lines 9-14) from a plurality of schedulers (e.g., other forms of schedulers, see page 23, line 15), where at least one of the plurality of schedulers (e.g., profiler as scheduler 32, see FIG. 2) selects processes (e.g., 34-36, see FIG. 2) to be run from a plurality of runable processes different from the plurality of schedulers and where executable code that chooses the second scheduler uses parameters that vary according to run time conditions (e.g., see page 22, lines 20-22 and page 23, lines 18-21). The software is recited as also including executable code, provided on a computer readable medium, that switches, during run time, from using the first scheduler to schedule tasks to using the second scheduler to schedule tasks (e.g., see page 22, lines 18-22). Claims 27-31, 33 and 34 depend directly or indirectly from independent claim 26.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 9-10, 16-17, 26-27 and 33-34 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Deng, et al., "Scheduling Real-Time Application in an Open Environment", IEEE, 1997, pages 308-319 (hereinafter "Deng").

2. Claims 1-8, 11-14, 18-20, 22-25 and 28-31 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Deng in view of U.S. Patent No. 5,630,130 to Perotto, et al. (hereinafter "Perotto").

ARGUMENT

I. The Examiner has failed to establish a prima-facie case of obviousness under 35 U.S.C. §103(a) of claims 9-10, 16-17, 26-27 and 33-34 as being unpatentable over Deng.

A. Obviousness Standard.

In determining whether or not there is a proper case of obviousness, it is necessary to establish whether one of ordinary skill in the art would, having the prior art references before him, be capable, or otherwise motivated, to make the proposed combination, modification or substitution so as to yield all elements of a claimed invention. *See KSR Int'l Corp. v. Teleflex Inc.*, 127 S. Ct. 1727, 82 USPQ2d 1385 (2007); *see also In re Lintner*, 458 F.2d 1013, 1016 (CCPA, 1972). In rejecting claims under 35 U.S.C. §103, it is incumbent upon the Examiner to establish a factual basis to support the legal conclusion of obviousness and the Examiner is expected to make the factual determinations set forth in *Graham v. John Deere Co.*, 383 U.S. 1, 17, 148 USPQ 459, 467 (1966). *See also United States v. Adams,* 383 U.S. 39 (1966); *Anderson's-Black Rock, Inc. v. Pavement Salvage Co.*, 396 U.S. 57 (1969); and *Sakraida v. AG Pro, Inc.*, 425 U.S. 273 (1976). The analysis used to combine prior art teachings to invalidate a patent claim based on obviousness should be explicitly articulated. *See KSR*, 82 USPQ2d at 1396, citing *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006) ("[R]ejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness"). However, the analysis may take account of the inferences and creative steps that a person of ordinary skill in the art would employ. Id.

B. Deng does not teach or fairly suggest every element of Appellants' claimed invention as to have rendered Appellants' claims 9-10, 16-17, 26-27 and 33-34 obvious to one of ordinary skill in the art at the time the invention was made.

As described above, all of Appellants' independent claims (and, accordingly, the claims depending therefrom) recite switching between a first scheduler and a second scheduler during run time and choosing the second scheduler from a plurality of schedulers based on parameters that vary according to run time conditions.

In the Final Office Action dated October 13, 2006 and the Advisory Action dated January 17, 2007, the Examiner states that Deng teaches multiple schedulers and cites to the OS scheduler, time sharing scheduler, RM-PCP scheduler and EDF-SBP scheduler of Deng's Figure 1. However, Appellants point out that a hierarchical scheduling scheme is disclosed in Deng's Figure 1, in which the OS scheduler is illustrated as part of the operating system and maintains all the servers in the system. Each server then has a scheduler associated with it, called a server scheduler (i.e., the time sharing scheduler, the RM-PCP scheduler, the EDF-SBP scheduler); however, the functionalities of the server schedulers are different than that of the OS scheduler. As further discussed herein, Appellants submit that the server schedulers do not disclose the switching between multiple schedulers during run time as is recited by Appellants.

Deng discloses a system whereby an application developer develops a real-time application, A_k, and then assigns a scheduling algorithm, $\sum_k$, to schedule jobs for A_k (see page 309, second column, bottom of Deng). As shown in Figure 1 and described in the

corresponding text, a single OS Scheduler (EDF) manages the various scheduling algorithms. As indicated at the bottom of column 2 on page 310, when a job of a real-time application A_k is released, the OS scheduler invokes the server scheduler of the server S_k to insert the newly released job in the proper location in the server's ready queue according to the scheduling algorithm $\sum_k$.

Appellants point out that Deng does not disclose switching between multiple schedulers during run time. Instead, as noted above, when a job of a real-time application is released, the single OS Scheduler (EDF) invokes a server scheduler to insert the newly-released job into the queue of that server scheduler. Each server scheduler corresponds to a different scheduling algorithm. Although multiple server schedulers are available to the OS Scheduler (EDF), Deng does not disclose switching between the server schedulers (or scheduler algorithms) once a particular server scheduler has been invoked for a job. Rather, Deng's invoked server scheduler processes the job within the queue of the server scheduler. As described by Deng at col. 310, right column, top, "Whenever a server is scheduled, it executes the job at the head of its ready queue."

In the Advisory Action, the Examiner cites to the 4th and 5th paragraphs of section 3.1 of Deng and, specifically, discusses processing of non-real time applications compared to processing of real-time applications. However, Deng discloses that non-real-time applications are always admitted by the system (and jobs of non-real time applications are processed by the time sharing server scheduler S_0 as shown in Fig. 1) but that real-time applications are admitted only upon passing an acceptance test. A new application requests admission by providing the OS Scheduler the information needed to conduct the acceptance

test. If the application passes the acceptance test, the system admits the application and processing proceeds as discussed above concerning a newly-released job of a real-time application involving processing by other server schedulers (e.g., RM-PCP server scheduler S_1 or EDF-SBP server scheduler S_2).

The Examiner appears to suggest that with respect to non-real-time applications, the OS Scheduler (EPF) would be a first scheduler processing non-real-time application tasks and that one of the real-time application server schedulers would be a second scheduler processing real-time application tasks. However, the OS Scheduler (EPF) does not "process" the non-real time applications tasks as suggested by the Examiner akin to the processing of the real-time application tasks by a server scheduler. Rather, the OS Scheduler (EPF) determines if an application is real-time or not and then chooses the appropriate server scheduler to process tasks of the applications. As discussed above, the time sharing server scheduler processes a job in the case of non-real-time applications and, for example, the RM-PCP or EDF-SBP server schedulers process a job in the case of real-time applications. The OS Scheduler invokes the appropriate server scheduler for processing a job but does not itself process the job as suggested by the Examiner. Accordingly, the interpretation of Deng proposed by the Examiner does not follow from the description of the functionalities of the OS Scheduler and server schedulers in Deng.

Basically, Deng discloses a using a single OS Scheduler. Although, as pointed out by the Examiner, the OS Scheduler may handle real time and non-real time applications differently (and may do other things based on application characteristics), the OS Scheduler of Deng does not swap itself, out and is not otherwise swapped out, based on run-time

conditions (or on anything else). As set forth in detail above, the single OS Scheduler of Deng chooses an appropriate server scheduler based on whether an application is real time or not real time, but the OS Scheduler is never swapped out in favor of any server scheduler or any other scheduler.

In contrast to Deng, Appellants recite switching from a first scheduler to a second scheduler during run time in a multitasking system for a processor. Specifically, during run time of a first scheduler to schedule tasks, a second scheduler is chosen from a plurality of schedulers based on parameters that vary according to run time conditions. Then, a switch is made during run time from using the first scheduler to schedule tasks to using the second scheduler to schedule tasks. Accordingly, Appellants respectfully submit that Deng does not teach or fairly suggest at least the above noted features as claimed by Appellants.

For the reasons noted above, it is requested that the Board reverse the Examiner's rejection under 35 U.S.C. 103(a).

II. The Examiner has failed to establish a prima-facie case of obviousness under 35 U.S.C. §103(a) of claims 1-8, 11-14, 18-20, 22-25 and 28-31 as being unpatentable over Deng in view of Perotto.

A. Obviousness Standard.

The standard for obviousness under 35 U.S.C. §103(a) is set forth above.

B. Deng and Perotto, taken alone or in any combination, do not teach or fairly suggest every element of Appellants' claimed invention as to have rendered Appellants' Claims 1-8, 11-14, 18-20, 22-25 and 28-31 obvious to one of ordinary skill in the art at the time the invention was made.

As noted above, all of Appellants' independent claims recite that during run time in which a first scheduler schedules tasks, a second scheduler is chosen from a plurality of schedulers based on parameters that vary according to run time conditions. A switch is made during run time from using the first scheduler to schedule tasks to using the second scheduler to schedule tasks. As recited in claims 1 and 18, during a context switch operation, a program counter is set to correspond to the code of the second scheduler. The processor then executes the code of the second scheduler at an address corresponding to the program counter, thereby indicating a switch from the first scheduler to the second scheduler.

Perotto discloses a multitasking controller having task storage means (2) for storing up to N tasks (P0,P1,P2,P3) where each task comprises a sequence of instructions. The controller also includes a microprocessor for processing, by time-sharing, a plurality of such N tasks, and a random access memory (12) for storing variable data created and used by said

microprocessor. The microprocessor further includes a scheduler (7) realized in hardware for controlling the use of the microprocessor or by such processes, and program counter storage means for storing N program counters (Pc0, Pc1, Pc2, Pc3) each for use by the scheduler (7), which is able select a different one of the program counters (Pc0, Pc1, Pc2, Pc3) when the task processed by the microprocessor is changed without the transfer of data from the random access memory (12).

The deficiencies of Deng discussed in the previous section are not overcome by the addition of Perotto, especially since Perotto discloses a *single* scheduler that schedules one of the four disclosed tasks (P0, P1, P2, P3). None of the tasks P0, P1, P2, or P3 of Perotto are themselves schedulers and, moreover, all of Appellants' independent claims specifically recite some form of the schedulers being different from the tasks scheduled by the schedulers. Since Perotto only discloses a single scheduler, Perotto (like Deng) cannot show, teach, or suggest at least the recited feature of the presently-claimed invention relating to switching between a first scheduler and a second scheduler during run time and choosing the second scheduler from a plurality of schedulers, where the choice of the second scheduler is based on parameters that vary according to run time conditions.

Accordingly, for the reasons noted above, it is requested that the Board reverse the Examiner's rejection under 35 U.S.C. 103(a).

CONCLUSION

In view of the above, it is respectfully requested that the Board reverse all of the Examiner's rejections under 35 U.S.C. 103(a).

Respectfully submitted,
MUIRHEAD AND SATURNELLI, LLC

Date: May 31, 2007

Donald W. Muirhead
Registration No. 33,978

Muirhead and Saturnelli, LLC
200 Friberg Parkway, Suite 1001
Westborough, MA 01581
T: (508) 898-8601
F: (508) 898-8602

CLAIMS APPENDIX

The claims on appeal are set forth below.

1. (Previously Presented) A method of switching during run time from a first scheduler to a second scheduler for a multitasking system for a processor, comprising:

choosing the second scheduler from a plurality of schedulers, wherein at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and wherein choosing the second scheduler is based on parameters that vary according to run time conditions;

setting, during a context switch operation, a program counter to an address corresponding to code of the second scheduler; and

the processor executing code of the second scheduler at an address corresponding to the program counter.

2. (Previously Presented) A method, according to claim 1, further comprising:

setting a stack pointer to an address corresponding to stack space for the second scheduler; and

the processor using the stack space at the stack pointer after executing code at the address corresponding to the program counter.

3. (Original) A method, according to claim 1, wherein all of the schedulers use the same stack.

4. (Cancelled)

5. (Previously Presented) A method, according to claim 1, wherein at least one of the schedulers is for statistical code profiling.

6. (Previously Presented) A method, according to claim 1, wherein the first scheduler is for start up conditions and the second scheduler is for steady state operation.

7. (Previously Presented) A method, according to claim 1, wherein choosing the second scheduler is performed by setting up a return from an exception that causes the scheduler to execute.

8. (Previously Presented) A method, according to claim 1, wherein setting a program counter includes modifying a variable that is modified according to the second scheduler.

9. (Previously Presented) A method of scheduling tasks in a multitasking operating system, comprising:

using a first scheduler to schedule tasks;

choosing a second scheduler from a plurality of schedulers, wherein at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and wherein choosing the second scheduler is based on parameters that vary according to run time conditions; and

switching, during run time, from using the first scheduler to schedule tasks to using the second scheduler to schedule tasks.

10. (Previously Presented) A method, according to claim 9, wherein choosing the second scheduler is performed by setting up a return from an exception that causes the second scheduler to execute.

11. (Previously Presented) A method, according to claim 9, wherein switching to using the second scheduler includes setting a program counter to an address corresponding to code of the second scheduler.

12. (Previously Presented) A method, according to claim 11, wherein setting a program counter includes modifying a variable that is modified according to the second scheduler.

13. (Previously Presented) A method, according to claim 9, further comprising:

setting a stack pointer to an address corresponding to stack space for the second scheduler; and

the processor using the stack space at the stack pointer after executing code at the address corresponding to the program counter.

14. (Original) A method, according to claim 9, wherein all of the schedulers use the same stack.

15. (Cancelled)

16. (Previously Presented) A method, according to claim 9, wherein at least one of the schedulers is for statistical code profiling.

17. (Previously Presented) A method, according to claim 9, wherein the first scheduler is for start up conditions and the second scheduler is for steady state operation.

18. (Previously Presented) Computer software in combination with a computer readable medium that switches, during run time, from a first scheduler to a second scheduler for a multitasking system for a processor, comprising:

executable code, provided on a computer readable medium, that chooses the second scheduler from a plurality of schedulers, wherein at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and wherein executable code that chooses the second scheduler uses parameters that vary according to run time conditions;

executable code, provided on a computer readable medium, that sets a program counter to an address corresponding to code of the second scheduler; and

executable code, provided on a computer readable medium, that causes the processor to execute code at an address corresponding to the program counter.

19. (Previously Presented) Computer software, according to claim 18, further comprising:

executable code, provided on a computer readable medium, that sets a stack pointer to an address corresponding to stack space for the second scheduler; and

executable code, provided on a computer readable medium, that causes the processor to use the stack space at the stack pointer after executing code at the address corresponding to the program counter.

20. (Previously Presented) Computer software, according to claim 18, wherein all of the schedulers use the same stack.

21. (Cancelled)

22. (Previously Presented) Computer software, according to claim 18, wherein at least one of the schedulers is for statistical code profiling.

23. (Previously Presented) Computer software, according to claim 18, wherein the first scheduler is for start up conditions and the second scheduler is for steady state operation.

24. (Previously Presented) Computer software, according to claim 18, wherein executable code that causes the processor to execute code at an address sets up a return from an exception that causes the second scheduler to execute.

25. (Previously Presented) Computer software, according to claim 18, wherein executable code that sets a program counter modifies a variable according to the second scheduler.

26. (Previously Presented) Computer software in combination with a computer readable medium that schedules tasks in a multitasking operating system, comprising:

executable code, provided on a computer readable medium, that uses a first scheduler to schedule tasks;

executable code, provided on a computer readable medium, that chooses a second scheduler from a plurality of schedulers, wherein at least one of the plurality of schedulers selects processes to be run from a plurality of runable processes different from the plurality of schedulers and wherein executable code that chooses the second scheduler uses parameters that vary according to run time conditions; and

executable code, provided on a computer readable medium, that switches, during run time, from using the first scheduler to schedule tasks to using the second scheduler to schedule tasks.

27. (Previously Presented) Computer software, according to claim 26, wherein executable code that chooses the second scheduler sets up a return from an exception that causes the scheduler to execute.

28. (Previously Presented) Computer software, according to claim 26, wherein executable code that switches to using the second scheduler sets a program counter to an address corresponding to code of the second scheduler.

29. (Previously Presented) Computer software, according to claim 28, wherein setting a program counter includes modifying a variable that is modified according to the second scheduler.

30. (Previously Presented) Computer software, according to claim 26, further comprising:

executable code, provided on a computer readable medium, that sets a stack pointer to an address corresponding to stack space for the second scheduler; and

executable code, provided on a computer readable medium, that causes the processor to use the stack space at the stack pointer after executing code at the address corresponding to the program counter.

31. (Previously Presented) Computer software, according to claim 26, wherein all of the schedulers use the same stack.

32. (Cancelled)

33. (Previously Presented) Computer software, according to claim 26, wherein at least one of the schedulers is for statistical code profiling.

34. (Previously Presented) Computer software, according to claim 26, wherein the first scheduler is for start up conditions and the second scheduler is for steady state operation.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.